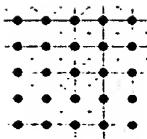
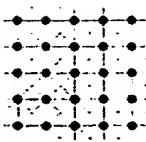


parimics



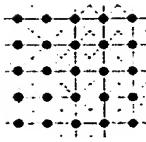
Parallel Image Computation System
Parallel versus Sequential Architecture

Parallel versus sequential



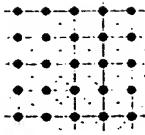
- MPP SIMDs like Parimics' IPE can access neighboring cells in a TDM fashion without conflict or contention.
- Sequential – even SMP – architectures need to go through a NorthBridge with an arbiter to access common (shared) memory.
- Memory is accessed using RAS and CAS as well as bank select. Only one entity can drive the address signals at any time, and that is the NorthBridge.

Slide Show explanation

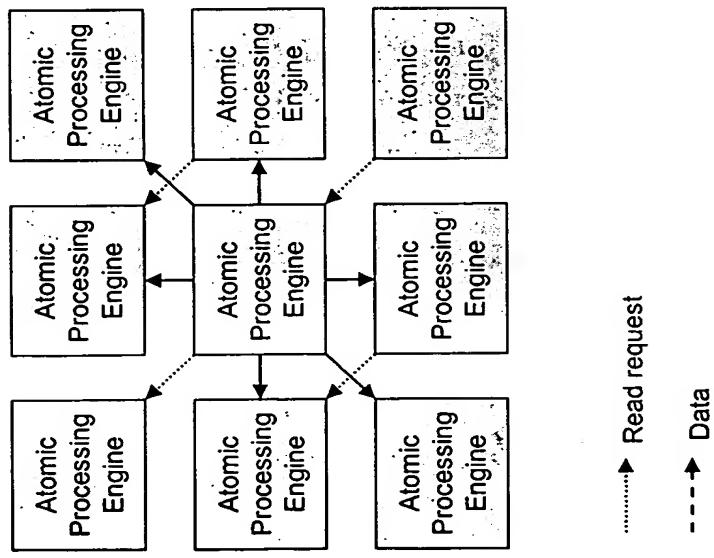


- The following slide show explains accesses of both an MPP SIMD to neighbors and an SMP into DRAM.
- The NorthBridge contains an arbiter and the DRAM controller.
- Arrows point in the direction of signal flow, be it address or data.
- Dotted arrows indicate read requests or address signals, dashed arrows indicate data signals.

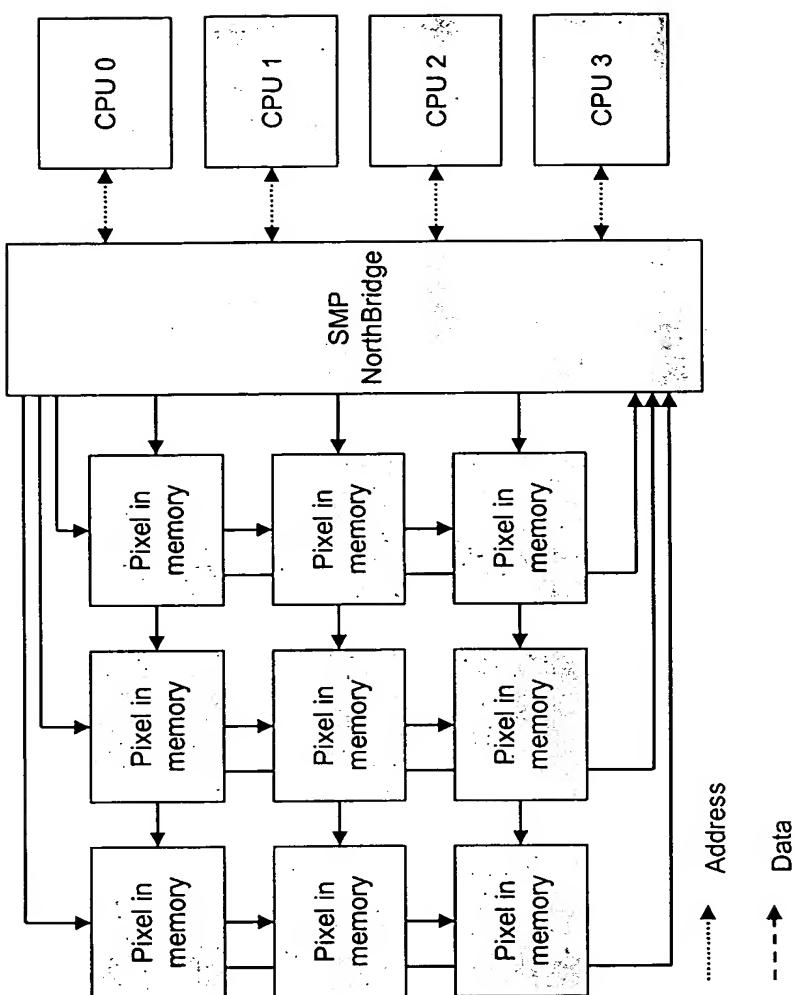
Cycle 0a



Parallel: Read Request 0

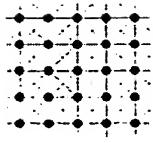


Sequential: Read Request CPUs to NorthBridge

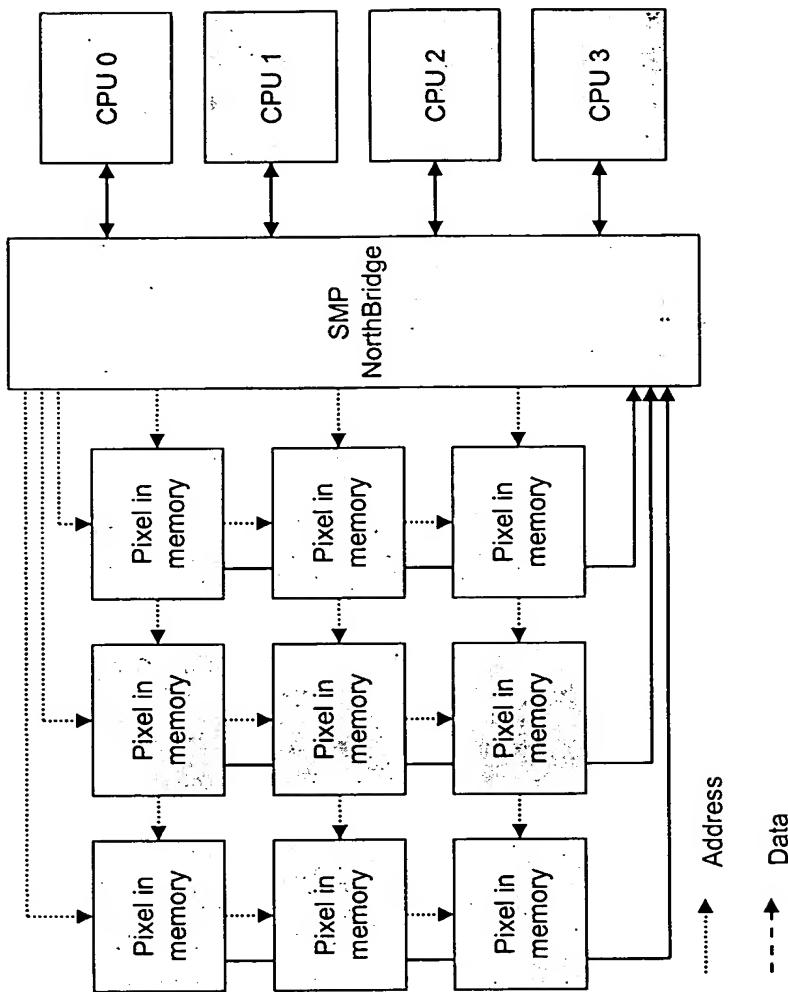


Cycle 0b

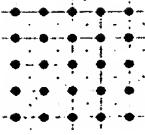
Parallel: Data 0



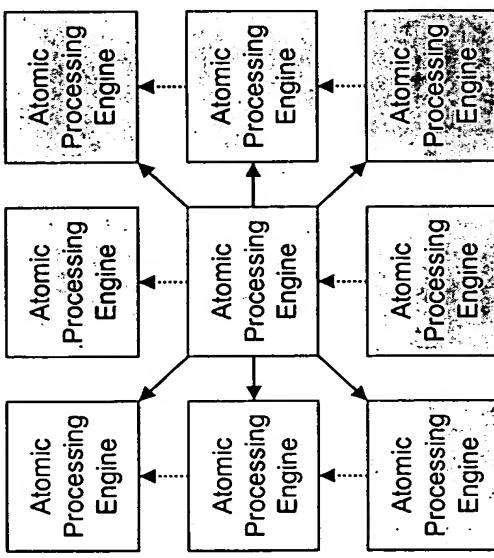
Sequential: RAS and CAS from NorthBridge to DRAM



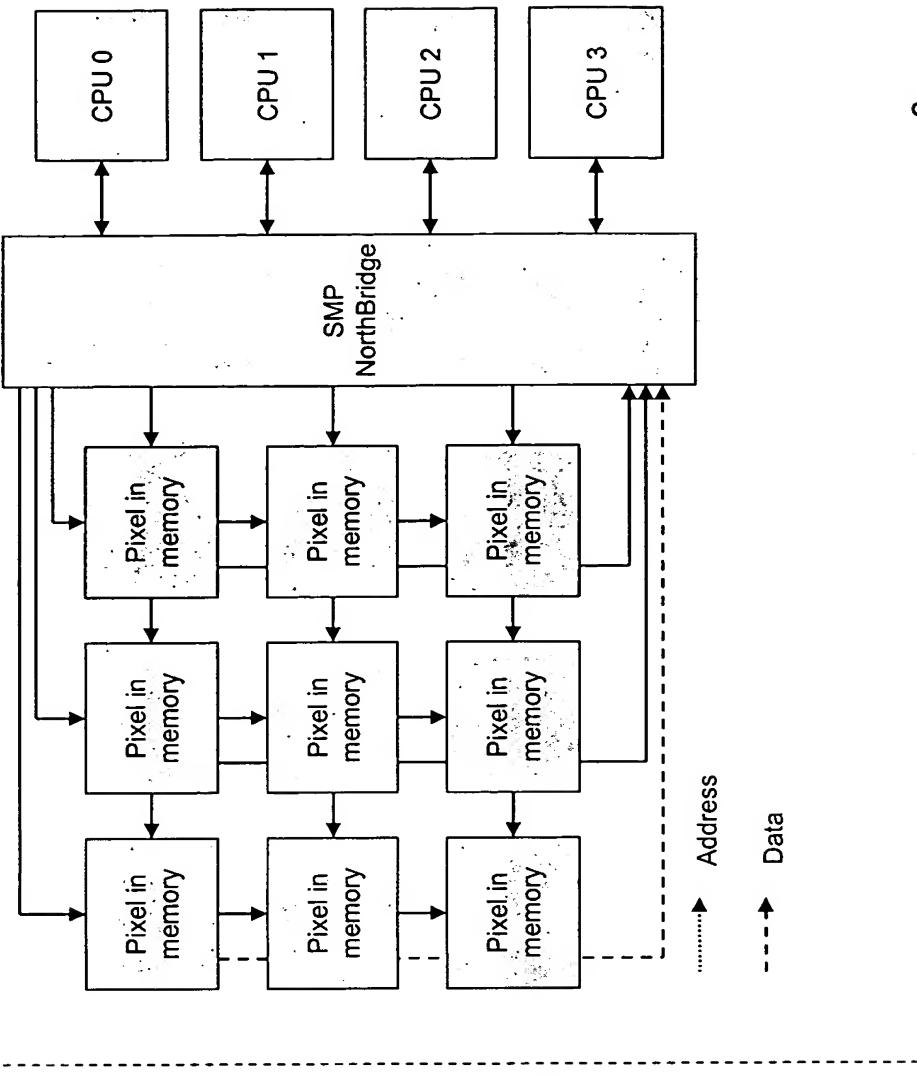
Cycle 1a



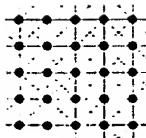
Parallel: Read Request 1



Sequential: Data out of DRAM cell 0

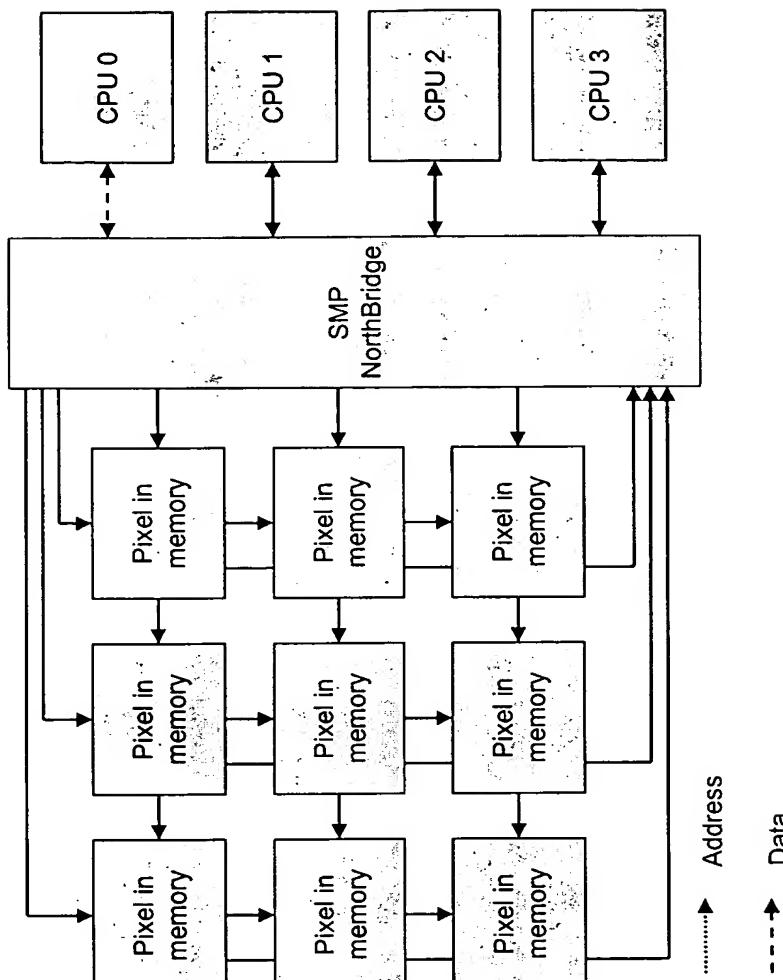


Cycle 1b



Parallel: Data out of

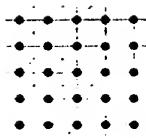
NorthBridge into CPU 0



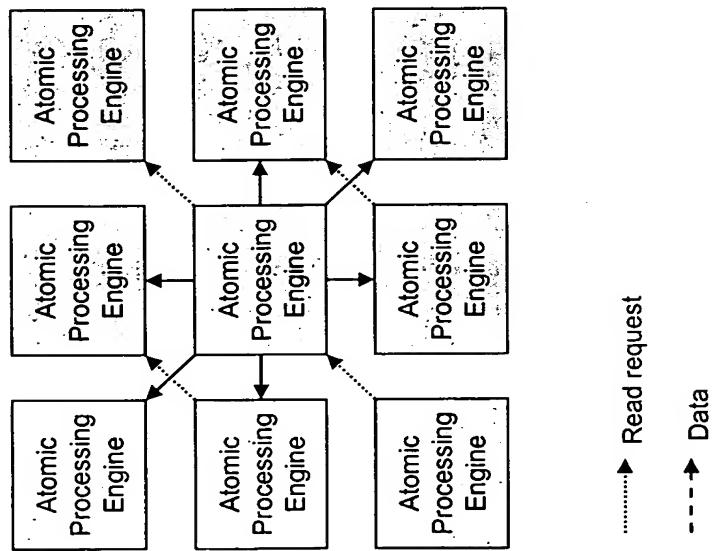
..... → Read request

- - - → Data

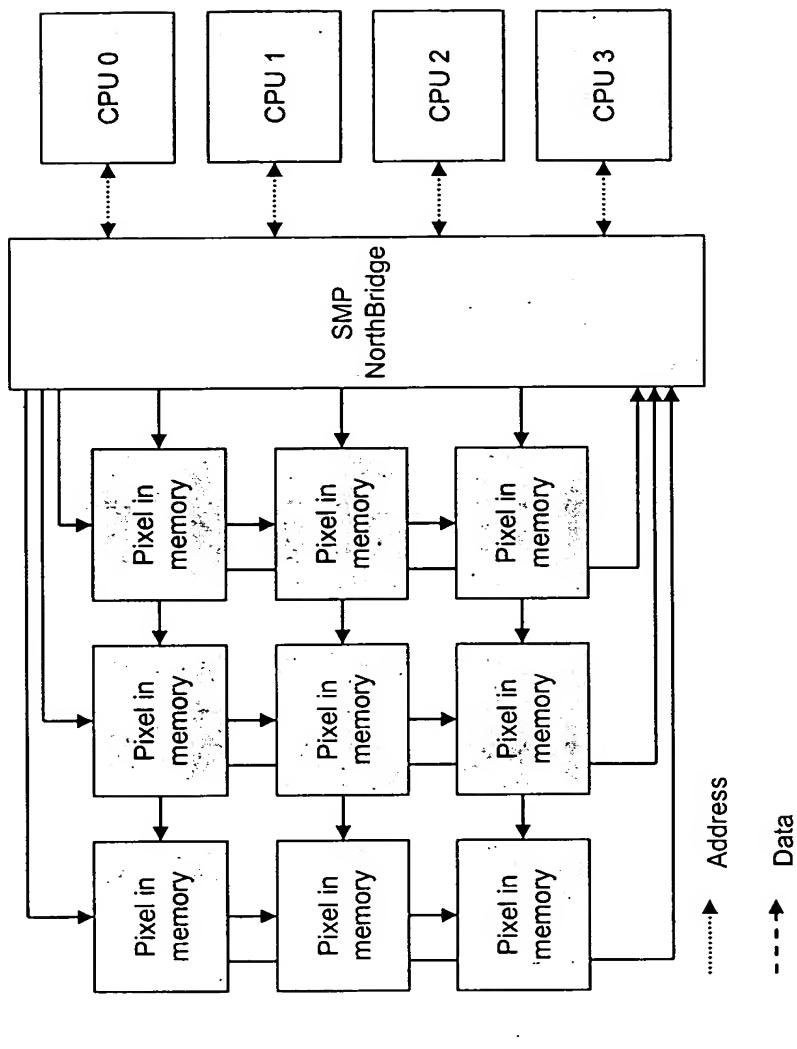
Cycle 2a



Parallel: Read Request 2

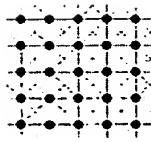


Sequential: Read Request CPUs to NorthBridge

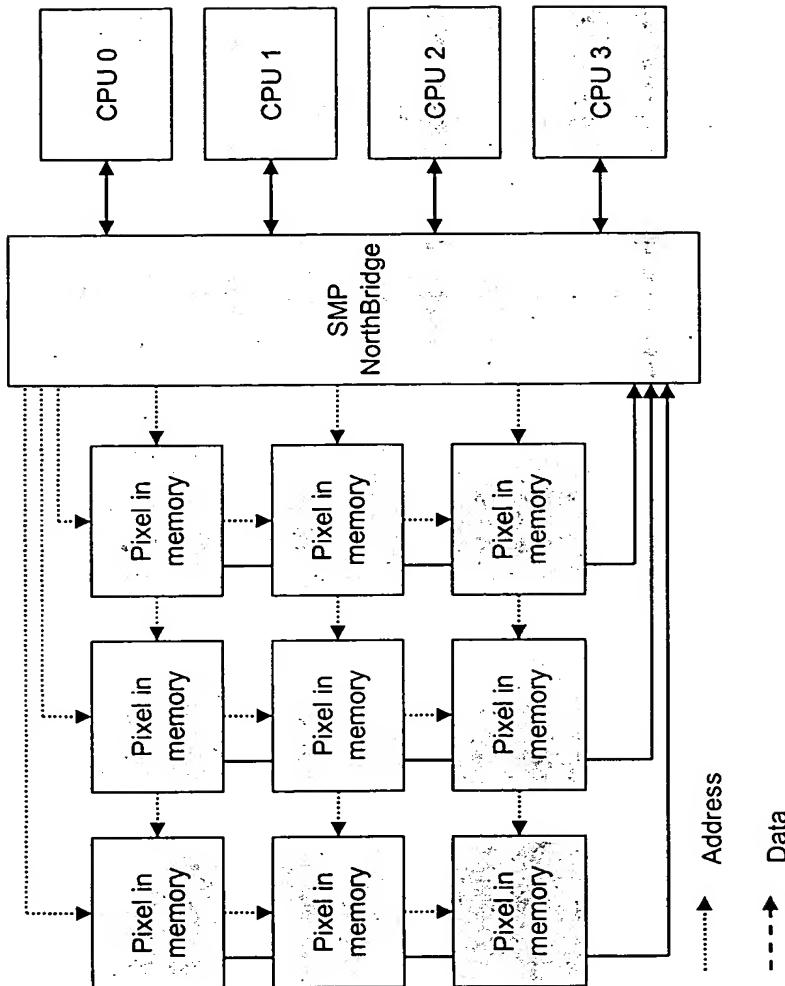


Cycle 2b

Parallel: Data 2



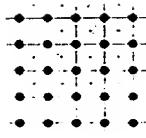
Sequential: RAS and CAS from NorthBridge to DRAM



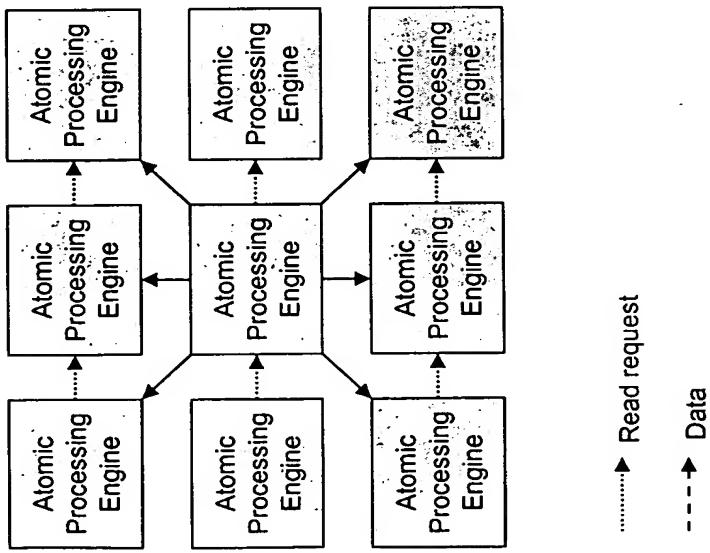
..... → Read request

- - - → Data

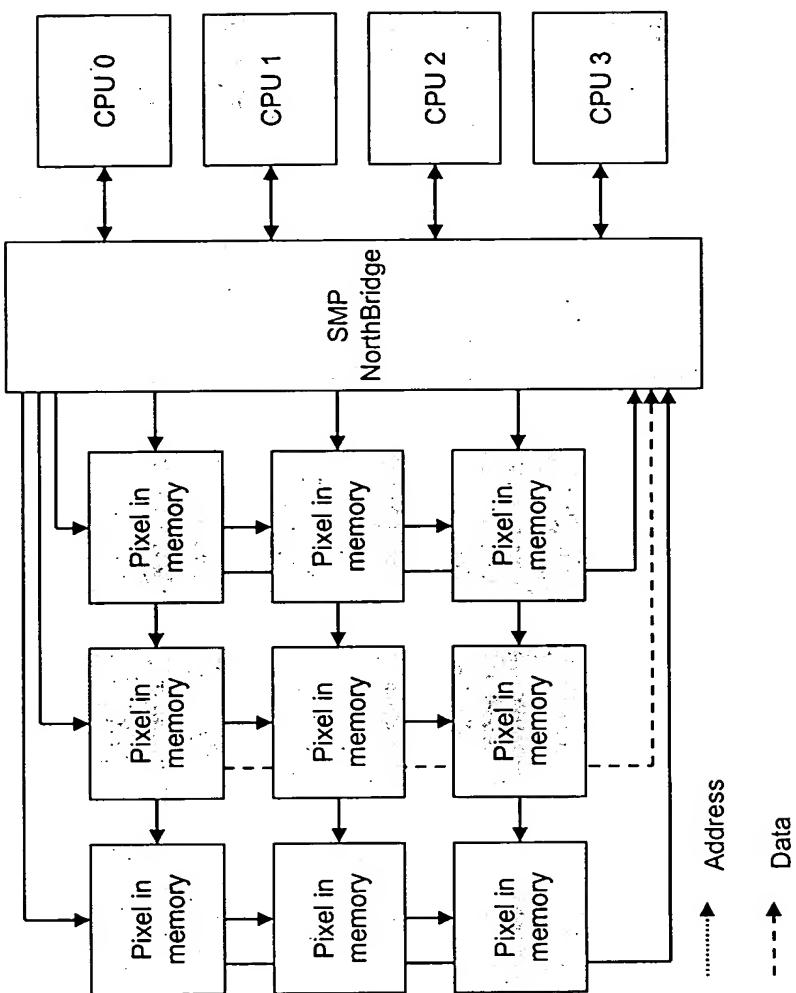
Cycle 3a



Parallel: Read Request 3

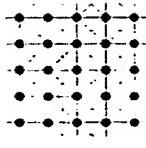


Sequential: Data out of DRAM cell 1

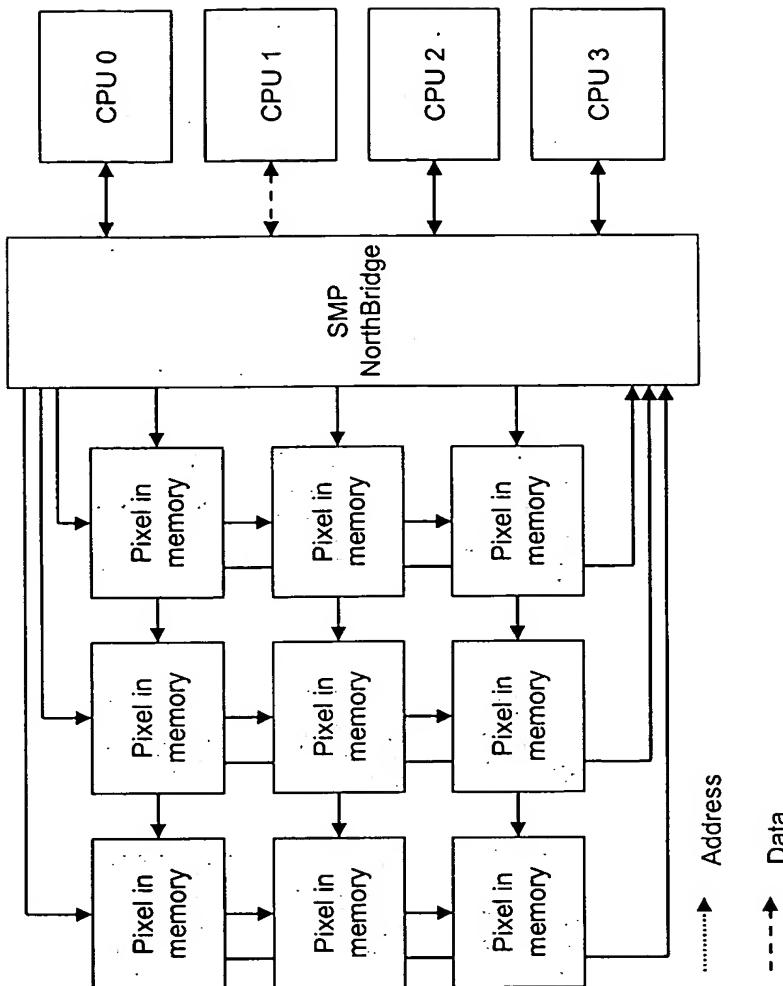


Cycle 3b

Parallel: Data 3



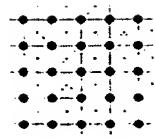
Sequential: Data out of NorthBridge into CPU 1



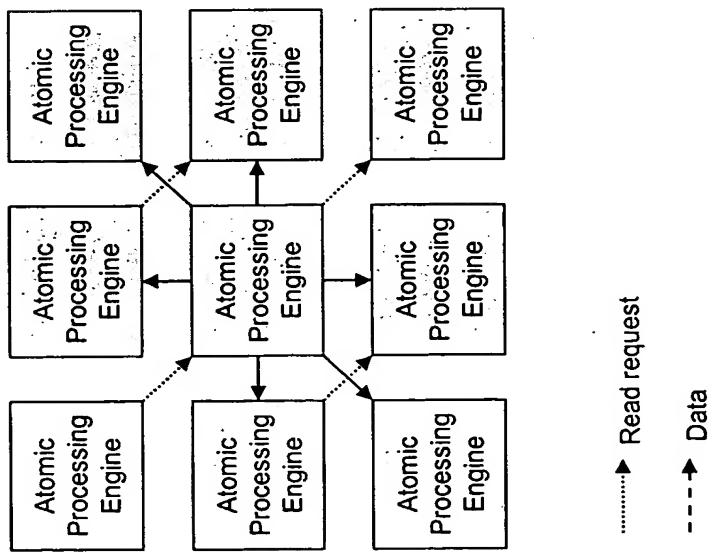
.....↑ Read request

- - - → Data

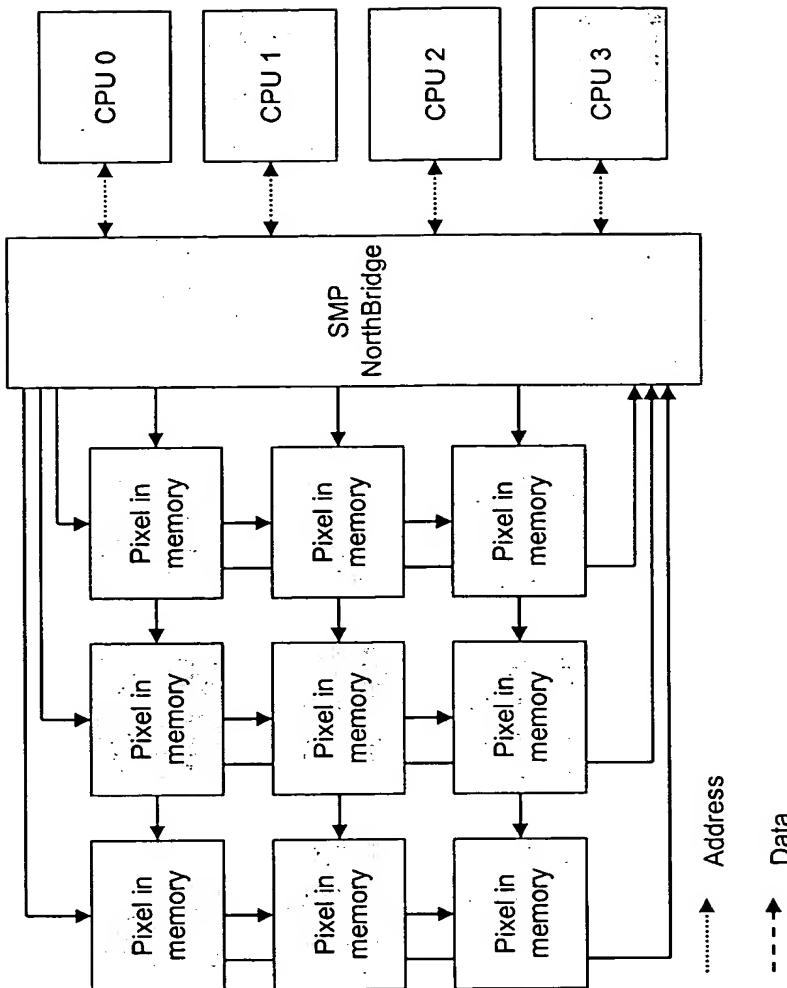
Cycle 4a



Parallel: Read Request 4

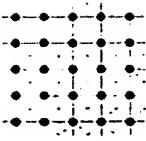


Sequential: Read Request CPUs to NorthBridge

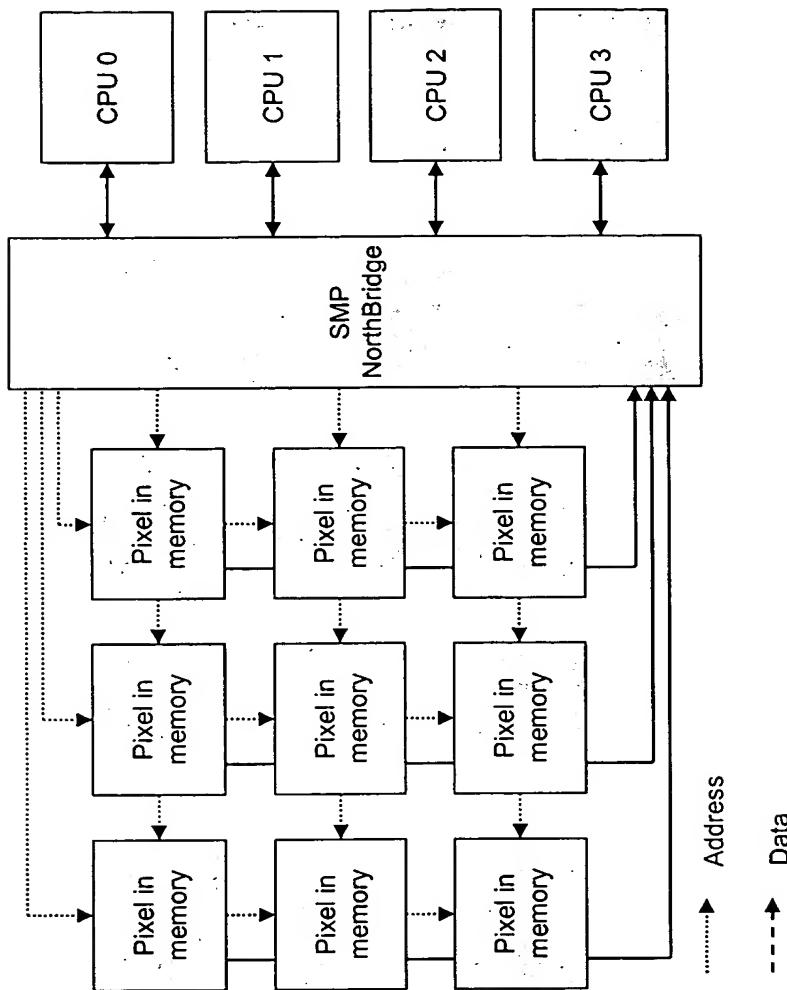


Cycle 4b

Parallel: Data 4



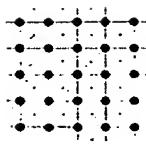
Sequential: RAS and CAS from NorthBridge to DRAM



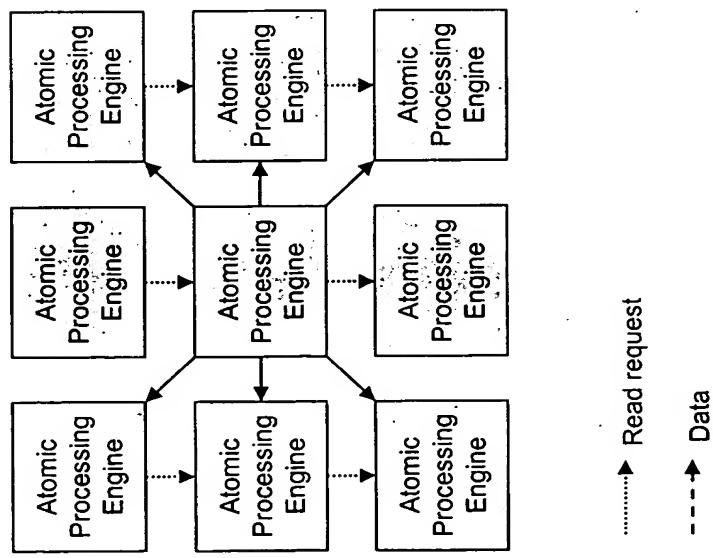
.....→ Read request

- - - → Data

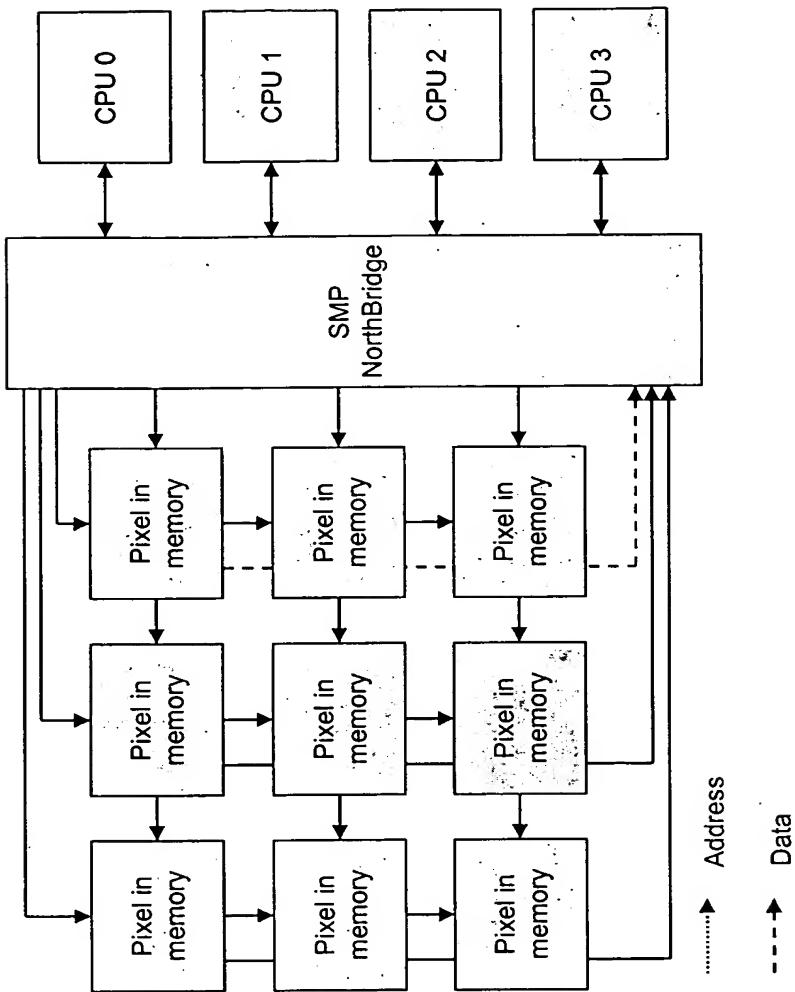
Cycle 5a



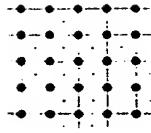
Parallel: Read Request 5



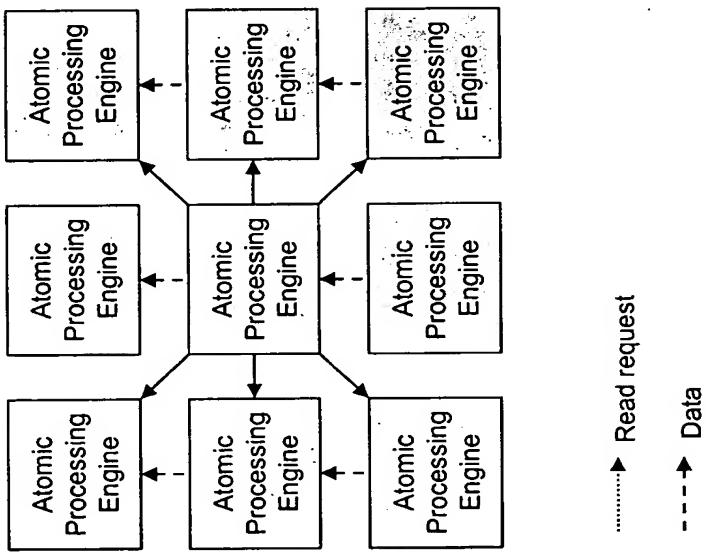
Sequential: Data out of DRAM cell 2



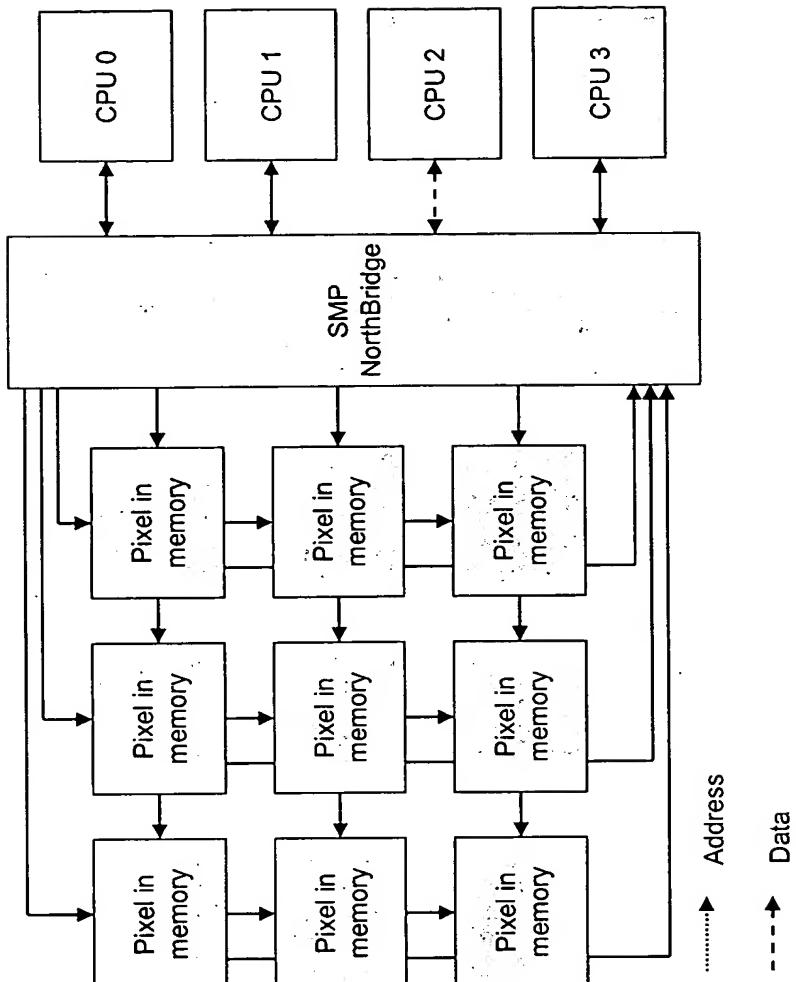
Cycle 5b



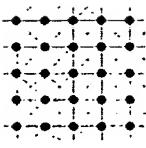
Parallel: Data 5



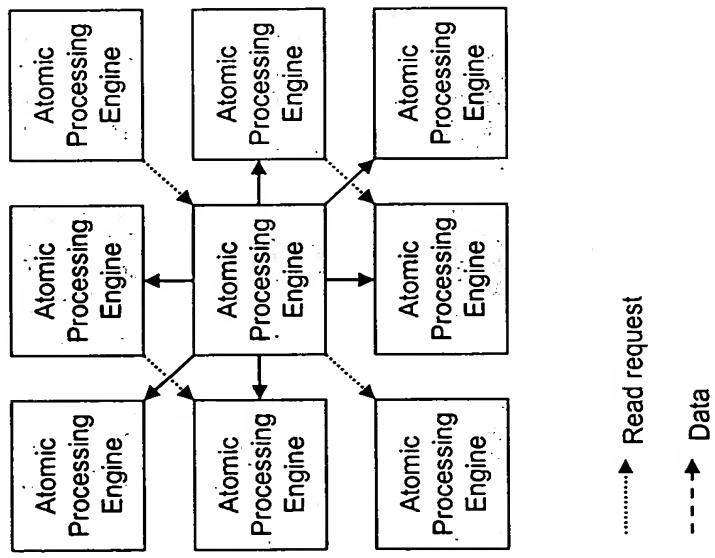
Sequential: Data out of NorthBridge into CPU 2



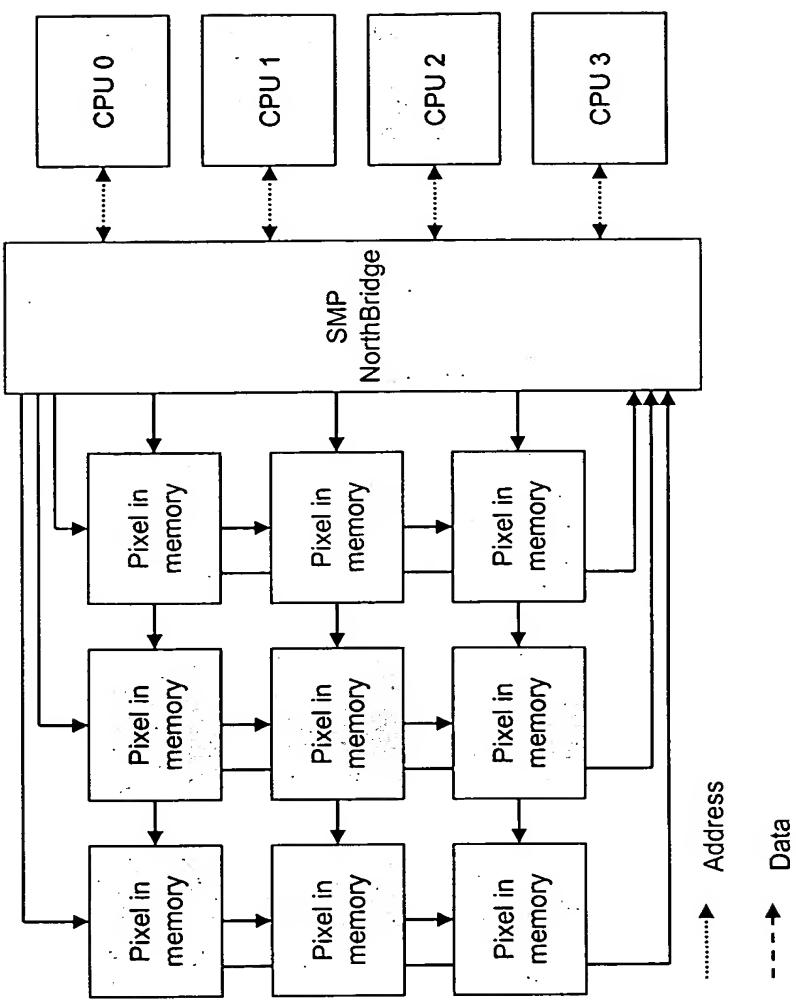
Cycle 6a



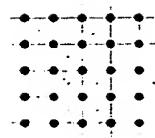
Parallel: Read Request 6



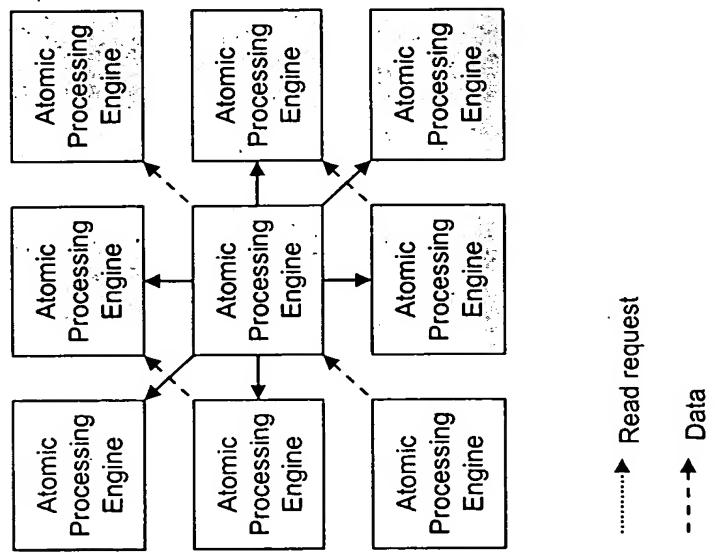
Sequential: Read Request CPUs to NorthBridge



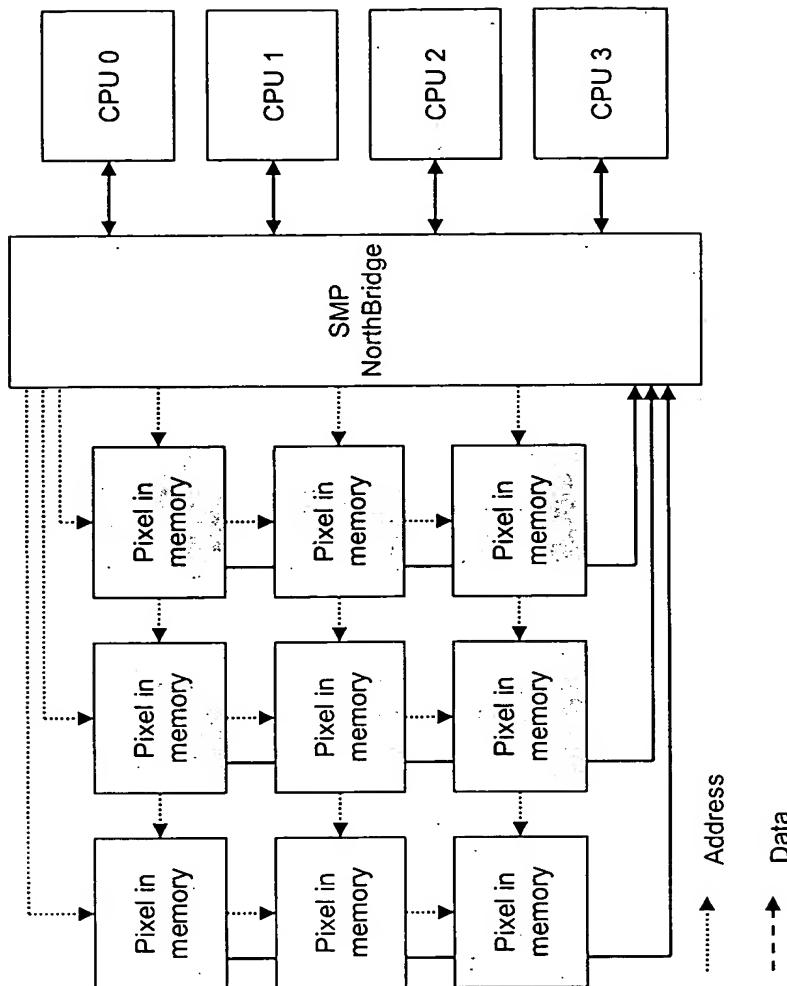
Cycle 6b



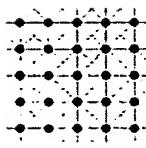
Parallel: Data 6



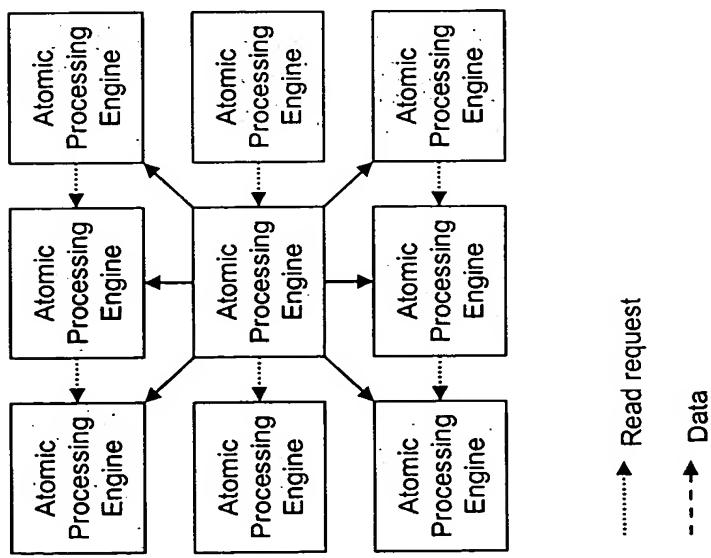
Sequential: RAS and CAS from NorthBridge to DRAM



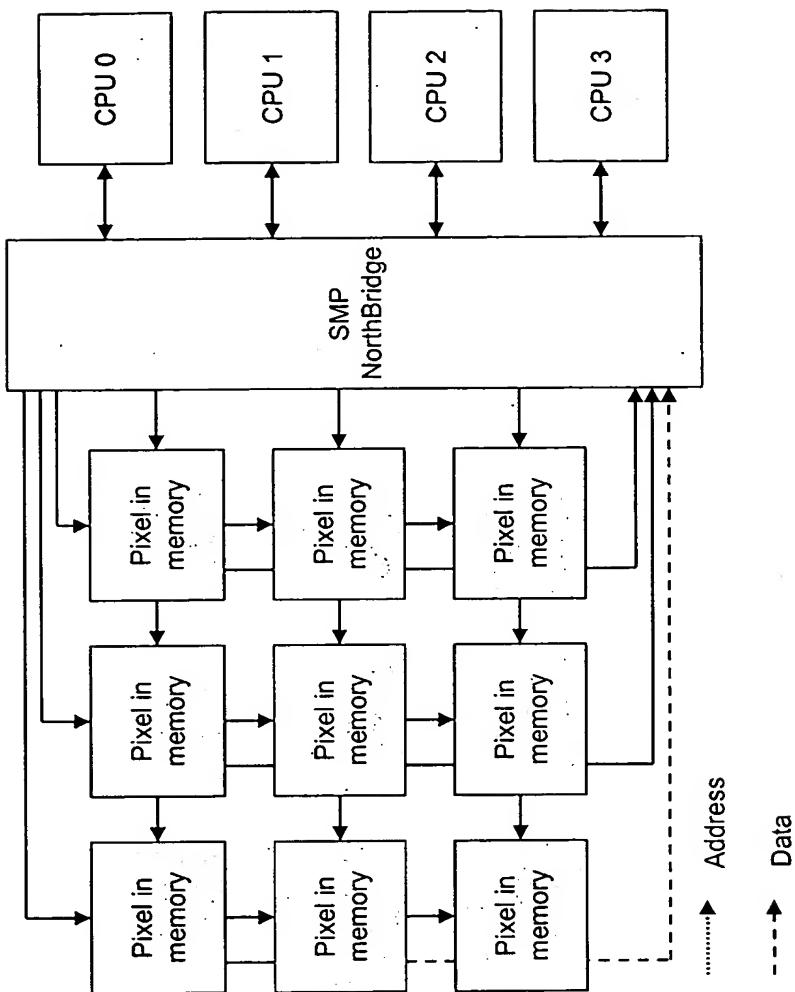
Cycle 7a



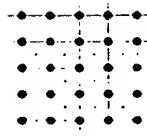
Parallel: Read Request 7



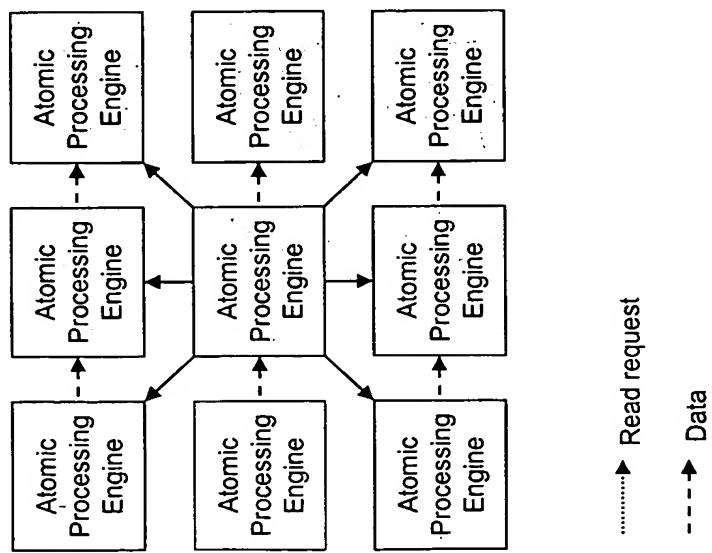
Sequential: Data out of DRAM cell 3



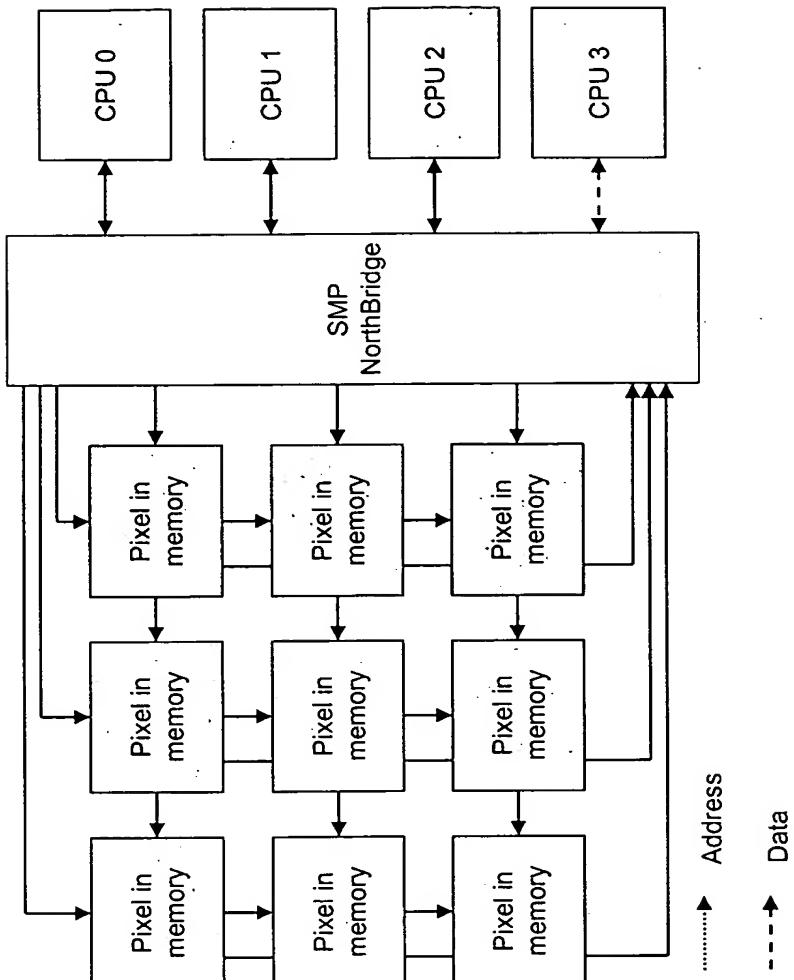
Cycle 7b



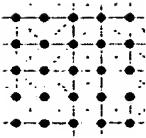
Parallel: Data 7



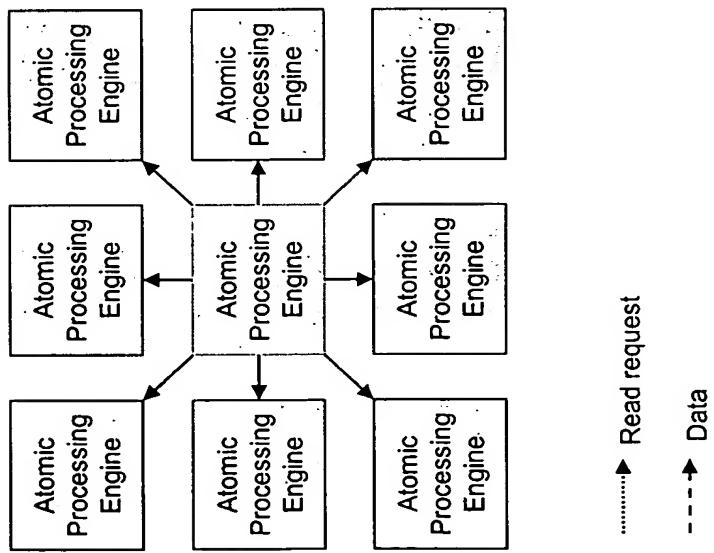
Sequential: Data out of NorthBridge into CPU 3



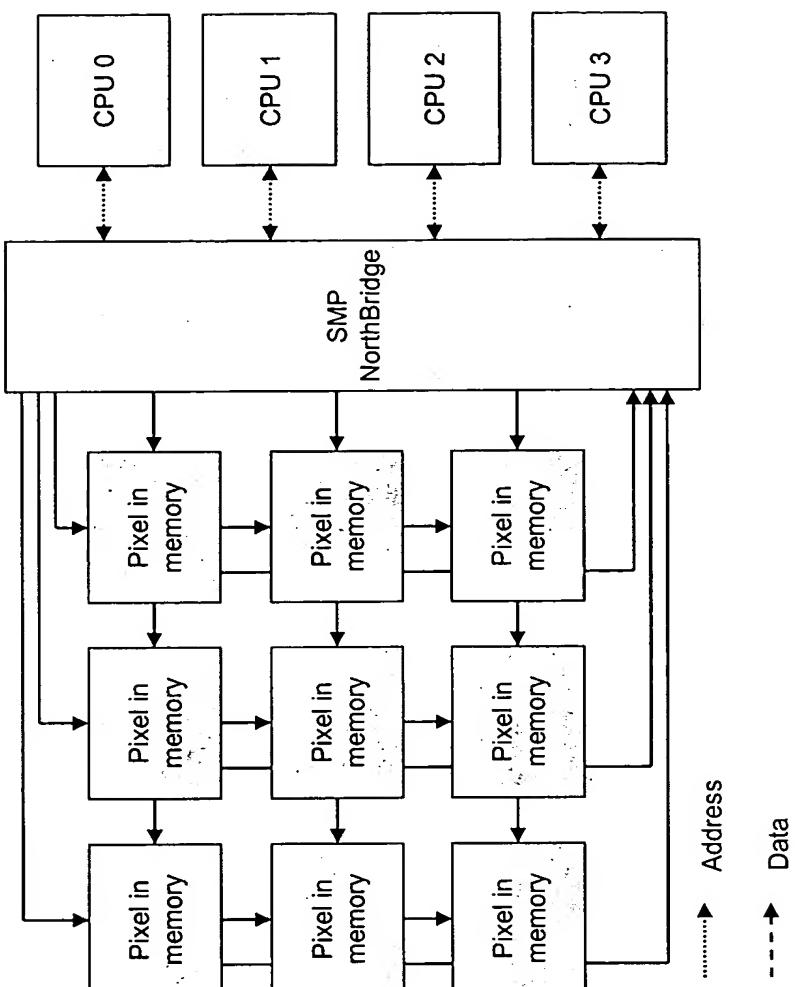
Cycle 8a



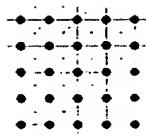
Parallel: Read Request 8 (int.)



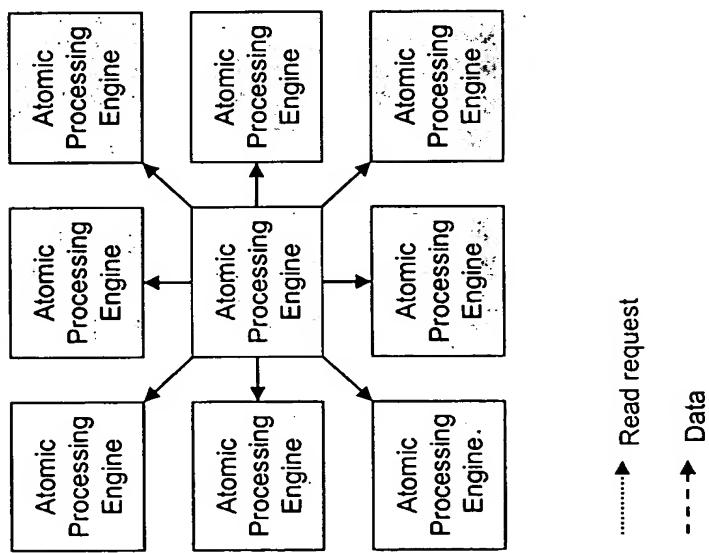
Sequential: Read Request CPUs to NorthBridge



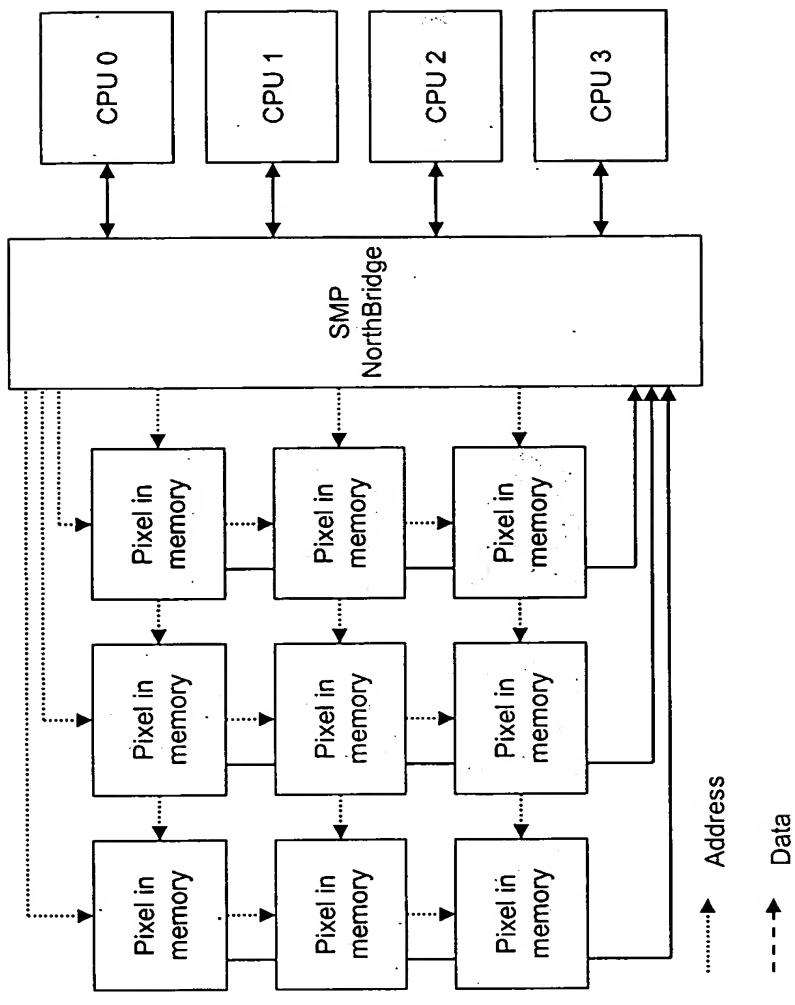
Cycle 8b



Parallel: Data 8 (int.)

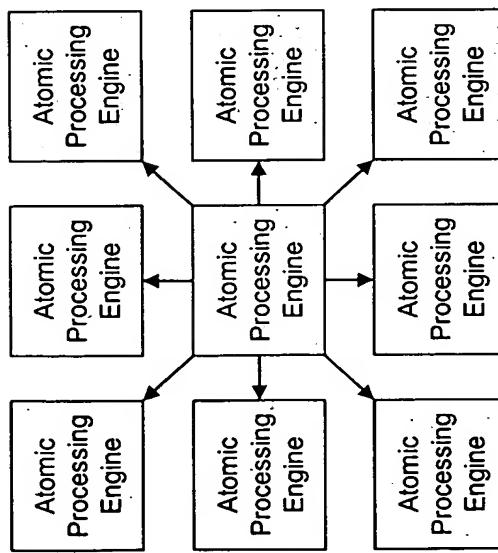


Sequential: RAS and CAS from NorthBridge to DRAM

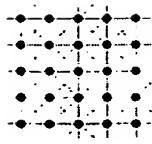


Cycle 9a

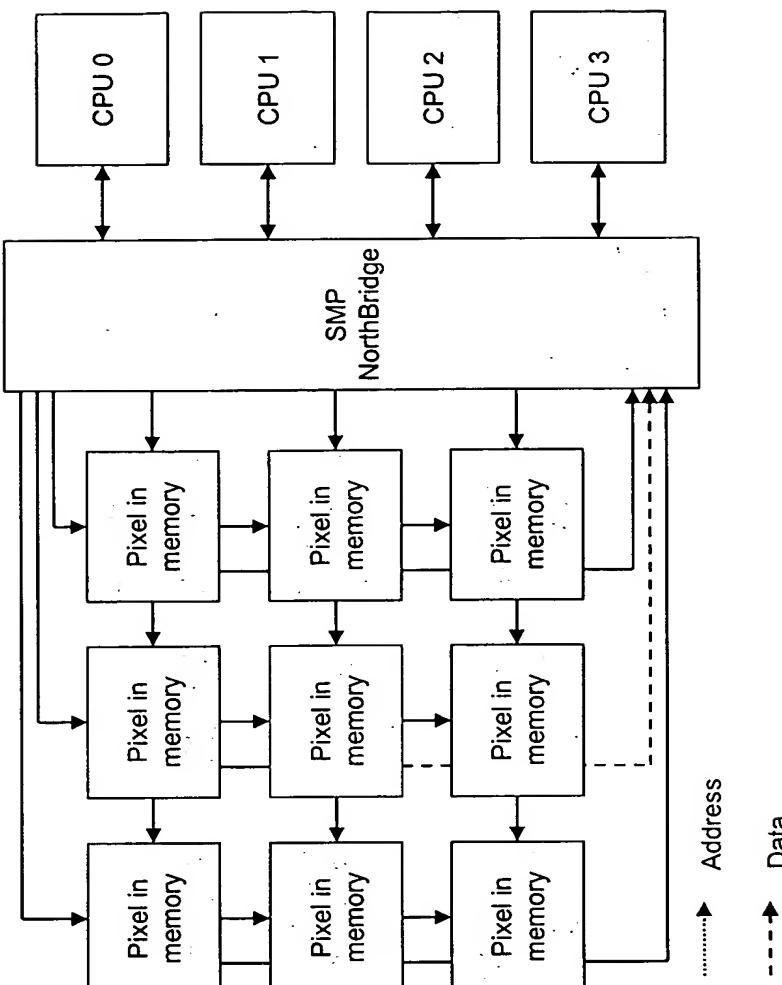
Parallel: Processing



.....→ Read request
- - - → Data

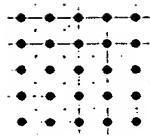


Sequential: Data out of DRAM cell 4

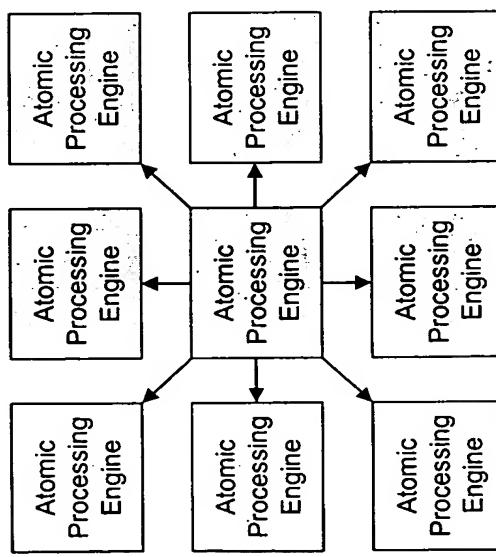


.....→ Address
- - - → Data

Cycle 9b



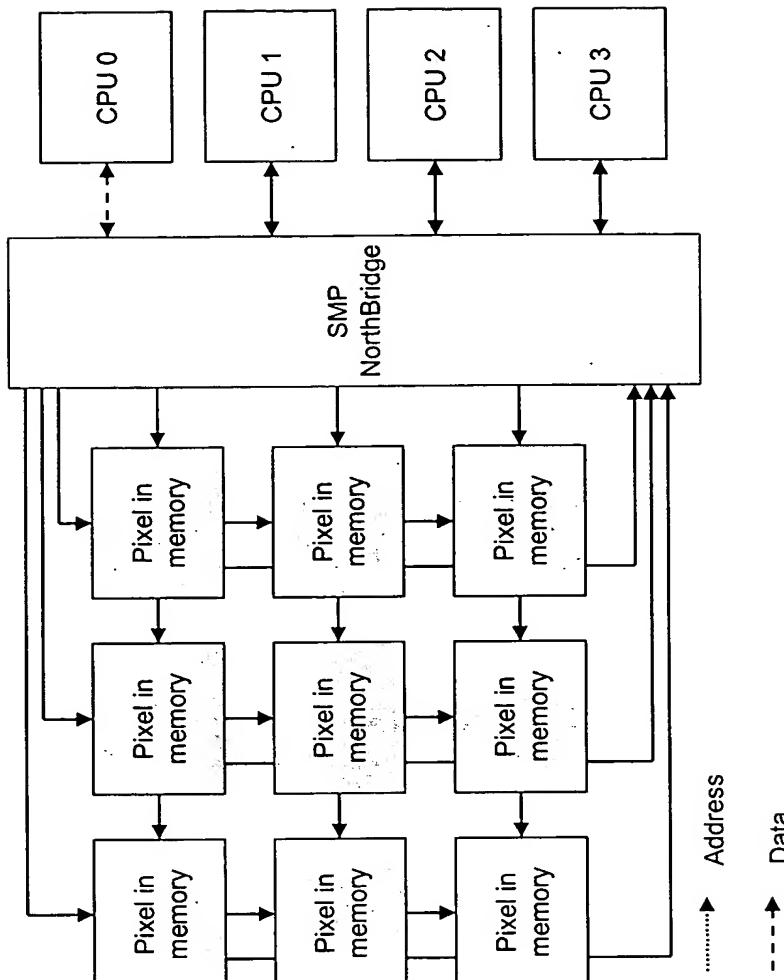
Parallel: Processing



.....→ Read request

- - - → Data

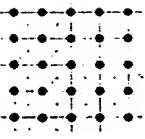
Sequential: Data out of NorthBridge into CPU 0



- - - → Address

.....→ Data

Conclusion



- At this point the center processor of the MPP SIMD cluster has polled all of its 8 neighbors and processed their data and its own. The new cell content is in the center processor. Since all processors are center processors of their own cluster, the MPP SIMD is done.
- The sequential architecture has finished polling 5 out of all - in this case 9 - cells, and the processing power not not used efficiently since mostly contention forced the processors to run idle.
- On a quad VGA resolution with 16 IPES, our parallel architecture would have finished the task, whereas a sequential processing architecture had completed 5 pixels out of $1280 * 960 = 1228800$.
- Uniprocessor CPU or DSP based architectures are memory bound in image processing applications, SMP systems are even more memory bound. They are not compute bound.